# Project Proposal

Derek Paulsen

Full text search engines are used in a plethora of applications. These search engines (such as Apache Lucene) are built for efficient retrieval of top-k documents based on a TF/IDF based scoring metric which is dot product between two sparse vectors $q^T d = score$[1][2][3]. While this default scoring gives decent results out of the box, it is frequently augmented by re-weighting the query $q$, with some weight vector $w$ changing to scoring to be $(q \odot w)^T d = score$. This allows for boosting of certain terms or fields to improve the quality of search results while not changing the underlying search algorithm.

The problem we will address in this project is finding a good weight vector $w$. In particular our problem setting is as follows. We are giving a set of query vectors $Q = \{q_1, ...q_n\}$. For each of these query vectors $q_i$ we are given a set of $k$ retrieved document vectors with labels $R_i = \{(d_{i,1}, l_{i,1}), ..., (d_{i,k}, l_{i,k})\}$, $l \in \{relevant, irrelevant\}$. We wish to find a weight vector $w$ that minimizes the number of irrelevant documents that are examined before the relevant result is found. This problem can be formulated as a mixed integer linear program (MILP) of the following form

$$\begin{aligned} \min_{w,z} \quad & 1^T z \\ s.t. \quad & Aw - \epsilon z \leq 0 \\ & w \geq 0 \\ & z \in \{0, 1\} \end{aligned}$$

Where $\epsilon$ is some large value that is used in conjunction with $z_j$ to indicate that some constraint has been violated. Solving MILP's is an NP-Hard problem and in practice is very computationally expensive, frequently requiring exponential time to solve. Because of this, many solutions have been proposed to reduce the search space examined for MILP's. The core of most of solvers is a branch and bound algorithm which solves LP relaxations of the problem and the branches into different sub-problems with fixed integer variables. Improvements to the core algorithm typically aim to provide better branching heuristics for branch and bound algorithms, such as strong branching [9]. More recent work has

examined machine learning for better branching heuristics [5][6][7][8], either to approximate the strong branching using machine learning (to reducing the computation cost) or to learn a completely new branching heuristic. In this project we plan to develop a new way of solving the above MILP. Instead of trying to create a branching heuristics we plan to leverage hardware accelerated gradient descent to approximate the optimal solution in polynomial time.

## References


1 - http://engineering.nyu.edu/~suel/papers/bmw.pdf

2 - https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.365.2939&rep=rep1&type=pdf

3 - https://cs.uwaterloo.ca/~jimmylin/publications/Grand_etal_ECIR2020_preprint.pdf

4 - http://cgi.di.uoa.gr/~sgk/teaching/grad/handouts/karp.pdf

5 - https://arxiv.org/pdf/1811.06128.pdf

6 - https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.705.606&rep=rep1&type=pdf

7 - https://arxiv.org/pdf/1307.4689.pdf

8 - https://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/download/12514/11657

9 - https://www.math.uwaterloo.ca/~bico/papers/tsp_icm.pdf